PATENT ABSTRACTS OF JAPAN

(11)Publication number : 07-200315
(43)Date of publication of application : 04.08.1995

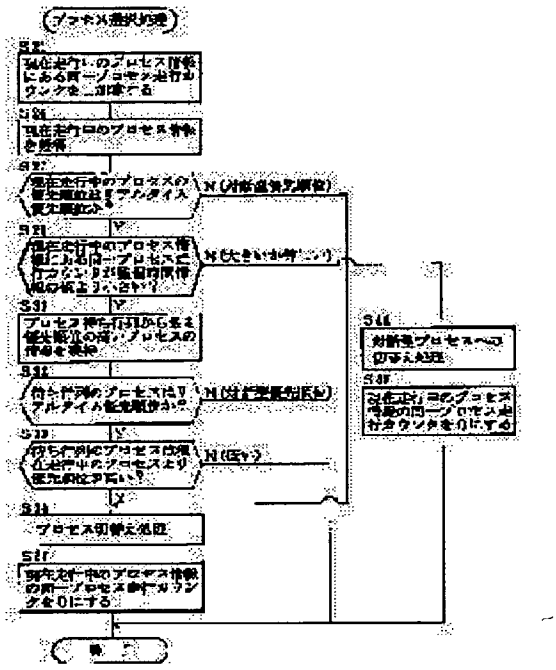
(51)Int.Cl. G06F 9/46

(21)Application number : 06-012101
(22)Date of filing : 07.01.1994

(71)Applicant : FUJI FACOM CORP
(72)Inventor : YAMAGUCHI HIROSHI

(54) SCHEDULING DEVICE FOR PROCESS IN COMPUTER SYSTEM

(57)Abstract:
PURPOSE: To attain proper scheduling by preventing a CPU from being continuously occupied by real time process.
CONSTITUTION: A same process running counter is incremented by 1 (S21), information relating to a process running at present is acquired from process information running at present (S26) to discriminate whether or not the process running at present has priority in real time. In the case of the priority of real time (S27Y), a count of a same process running counter in the process information running at present is compared with monitor time information and when the count of the same process running counter is larger than or equal to the monitor time information value (S28N), an interactive process changeover module is executed (S44), and the same process running counter in the process information running at present is set to zero and the program is terminated (S47).



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平7－200315

(43)公開日　平成７年(1995)８月４日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/46	３４０　Ｂ	7629－5B		

審査請求　未請求　請求項の数６　ＦＤ　（全 15 頁）

(21)出願番号　特願平6－12101

(22)出願日　平成６年(1994)１月７日

(71)出願人　000237156
富士ファコム制御株式会社
東京都日野市富士町１番地

(72)発明者　山口　弘
東京都日野市富士町１番地　富士ファコムシステム株式会社内

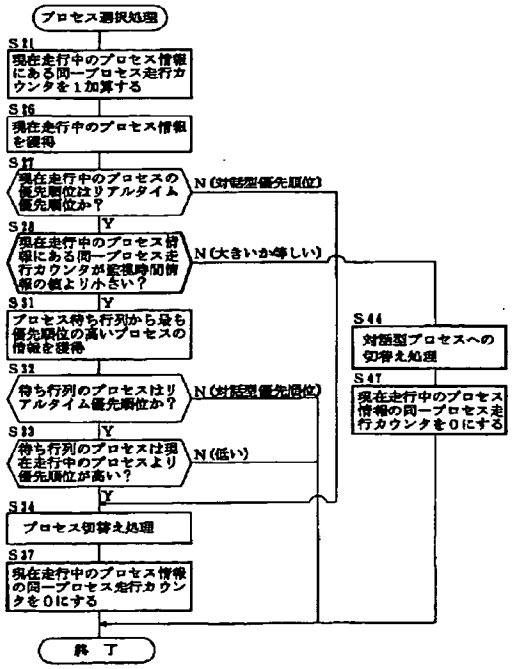
(74)代理人　弁理士　森田　雄一

(54)【発明の名称】　計算機システムにおけるプロセスのスケジューリング装置

(57)【要約】

【目的】　リアルタイムプロセスによりＣＰＵが連続して占有されることを防ぎスケジューリングを適正にする。

【構成】　同一プロセス走行カウンタに１を加算してから（Ｓ２１）、現在走行中のプロセスに関する情報を現在走行中プロセス情報から獲得し（Ｓ２６）、現在走行中のプロセスがリアルタイムの優先順位か否かを判別する。リアルタイムの優先順位である場合は（Ｓ２７Ｙ）、さらに、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、同一プロセス走行カウンタの値が監視時間情報の値よりも大きいか等しい場合は（Ｓ２８Ｎ）、対話型プロセス切替えモジュールを実行し（Ｓ４４）、現在走行中プロセス情報の同一プロセス走行カウンタを０にして終了する（Ｓ４７）。



1

【特許請求の範囲】

【請求項１】　プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

同一のリアルタイムプロセスが連続してＣＰＵを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したらプロセス待ち行列にリンクされている対話型プロセスの中の優先順位が最高の対話型プロセスにＣＰＵ占有を割り当てる手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項２】　プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

ユーザにより作成されたハンドラプログラムを予め格納した記憶手段と、

同一のリアルタイムプロセスが連続してＣＰＵを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したら記憶手段のハンドラプログラムによりスケジューリングを実行する手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項３】　プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

同一のリアルタイムプロセスが連続してＣＰＵを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したら現在走行中のリアルタイムプロセスを強制終了するとともにプロセス待ち行列のリンクから外す手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項４】　プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

同一のリアルタイムプロセスが連続してＣＰＵを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したら現在走行中のリアルタイムプロセスとプロセス待ち行列にリンクされている次の優先順位のリアルタイムプロセスとの間

2

で互いの優先順位を入れ換える手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項５】　プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

リアルタイムクラスのプロセスが連続してＣＰＵを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したらプロセス待ち行列にリンクされている対話型クラスの中の優先順位が最高の対話型プロセスにＣＰＵ占有を割り当てる手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【請求項６】　プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、

ユーザにより作成されたハンドラプログラムを予め格納した記憶手段と、

リアルタイムクラスのプロセスが連続してＣＰＵを占有した回数をカウントする手段と、

カウント手段のカウント値が所定値に達したら記憶手段のハンドラプログラムによりスケジューリングを実行する手段と、

を備えたことを特徴とする計算機システムにおけるプロセスのスケジューリング装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、１つのオペレーティングシステムで、対話型プロセスとリアルタイムプロセスに対し、定周期タイマ割込みによりＣＰＵを割り当てる際のスケジューリング装置に関する。

【０００２】

【従来の技術】従来、計算機システムにおいて、１つのオペレーティングシステム上に、リアルタイムプロセスと対話型プロセスとが混在することがある。ここでいうリアルタイムプロセスは、応答時間が厳しく制限されたプロセスであり、所定の時間内に処理が完了するため処理時間を予測することが可能である。また、対話型プロセスは、応答時間の制限は厳しくないが、どのプロセスにも均等にＣＰＵが割当てられる。

【０００３】さらに詳しく述べると、リアルタイムプロセスが複数のプロセスにより構成される場合、それぞれのプロセスはＣＰＵを割り当てるための優先順位が定められており、優先順位が高いプロセスから順にＣＰＵの占有権を獲得する。また、ＣＰＵを占有したプロセスは

処理が完了するまで占有権を放棄しない。すなわち、あるプロセスがＣＰＵを占有している間は、このプロセスより優先順位が下位のプロセスはＣＰＵの割当てを待つ状態を続ける。しかし、優先順位が高いプロセスに処理要求があると、そのプロセスはＣＰＵを占有しているプロセスからＣＰＵの占有権を奪い取り、自身がＣＰＵを占有する。このようにして、優先順位が高いプロセスから順に処理される。なお、この優先順位は、ユーザにより指定されるものである。

【０００４】一方、対話型プロセスは、オペレーティングシステムによって各プロセスの優先順位が決められる。すなわち、計算機システムの起動時は、各プロセスの優先順位はどれも同じであるが、各プロセスがＣＰＵを占有するごとに、そのプロセスの優先順位が低位に変更される。それにより、ＣＰＵの占有回数の少ないプロセスの優先順位が相対的に高くなり、順に最高となったプロセスからＣＰＵを占有することになる。その結果、平均すると、各プロセスに対してＣＰＵの割当て時間が均等になる。

【０００５】なお、このような対話型プロセスを有する計算機システムでは、一定間隔ごとの定周期タイマ割込みがオペレーティングシステムとしておこなわれ、この割込み間隔を１単位時間とすると、対話型プロセスは１単位時間でＣＰＵの割当てが切り換わる。すなわち、ある対話型プロセスが１単位時間ＣＰＵを占有したら、そのプロセスは優先順位を低位に変更したのちＣＰＵの占有権を放棄する。次いで、その次の優先順位の対話型プロセスにＣＰＵの占有権が引き渡される。これらのリアルタイムプロセスと対話型プロセスとが１つのオペレーティングシステム上に混在する計算機システムにおけるスケジューリング方法は、次のように行われている。

【０００６】リアルタイムプロセスと対話型プロセスには、両プロセスごとに異なる優先順位を持たせる。例えば、システムに１２８の優先順位がある場合に、優先順位０～６３をリアルタイムプロセスとし、残りの優先順位６４～１２７を対話型プロセスとする。この優先順位は数字が小さい程、優先順が高くなる。また、実行可能であってＣＰＵが割当てられてないプロセスをリンクしておくために、プロセスの優先順位ごとに待ち行列を設ける。それにより、ＣＰＵを占有をしていたプロセスが占有権を放棄した場合、この待ち行列のなかの優先順位が最高位のプロセスがＣＰＵの占有権を獲得する。なお、優先順位の同じプロセスが複数個あった場合は、到着順（ＦＣＦＳ：Ｆｉｒｓｔ　Ｃｏｍｅ　Ｆｉｒｓｔ　Ｓｅｒｖｉｃｅ）によりＣＰＵの占有権を獲得する。

【０００７】リアルタイムプロセスと対話型プロセスとの間でのＣＰＵの占有については、リアルタイムプロセスを優先し、全てのリアルタイムプロセスの処理が完了してから、対話型プロセスにＣＰＵの占有権を与える。また、対話型プロセスがＣＰＵの占有権を得ている間であっても、新たにリアルタイムプロセスの処理要求が発生すると、対話型プロセスのＣＰＵ占有権は、リアルタイムプロセスに奪い取られる。これらの処理を行う従来のオペレーティングシステムの構成を示したのが図１４の概念図であり、スケジューリング処理を系統的に示したのが図１１、図１２、図１３のフローチャートである。

【０００８】図１４に示すように、オペレーティングシステムはタイマ割込みが処理モジュール、プロセス選択処理モジュール、プロセス切替え処理モジュール、プロセス待ち行列および現在走行中のプロセス情報から構成されており、定周期タイマ割込みが発生すると、タイマ割込み処理が呼び出される。図１１は、タイマ割込み処理モジュールを示す。このモジュールは、先ず、所定のタイマ処理を行い、次にプロセス選択処理モジュールへ進む。

【０００９】図１２は、プロセス選択処理モジュールの内容を示す。このモジュールは、先ず、現在走行中のプロセスに関する情報を獲得し（Ｓ２６）、次いで、現在走行中のプロセスがリアルタイムの優先順位か否かを判別する（Ｓ２７）。リアルタイムの優先順位である場合は（Ｓ２７Ｙ）、プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得する（Ｓ３１）。次に、待ち行列のプロセスはリアルタイム優先順位か否かを判別し（Ｓ３２）、リアルタイム優先順位である場合（Ｓ３２Ｙ）は、さらに、待ち行列のプロセスは現在走行中のプロセスより優先順位が高いか否かを判別する（Ｓ３３）。

【００１０】優先順位が高い場合（Ｓ３３Ｙ）、すなわち図１４のプロセス行列待ちに、現在走行中のプロセスより優先順位の高いリアルタイムプロセスがリンクされていたらプロセス切換え処理モジュール（Ｓ３４）へ進む。待ち行列のプロセスがリアルタイム優先順位でなくて対話型優先順位である場合（Ｓ３２Ｎ）および優先順位が高くなくて低い場合（Ｓ３３Ｎ）は、現在走行中のプロセスがそのままＣＰＵの占有を続ける。また、Ｓ２７でリアルタイム優先順位でない場合も、プロセス切換え処理モジュール（Ｓ３４）へ進む。

【００１１】図１３は、プロセス切替え処理モジュールの内容を示す。このモジュールは、先ず、プロセス待ち行列に現在走行中のプロセスをリンクする（Ｓ１１）。なお、ここで現在走行中のプロセスが対話型プロセスだったら、優先順位を再計算してからリンクする。これは、対話型プロセスはＣＰＵを占有するごとに優先順位を下げることで、ＣＰＵの割当ての少ないプロセスの優先順位を相対的に高めるためである。次に、プロセス待ち行列から最も優先順位が高いプロセスを選びだしプロセス待ち行列のリンクから外し（Ｓ１２）、ＣＰＵを割り当てる（Ｓ１３）。最後に、現在走行中のプロセス情報（図１４）にＣＰＵを新たに割当てたプロセスの識別

子と優先順位を設定する（Ｓ１４）。

【００１２】

【発明が解決しようとする課題】しかしながら、上述したオペレーティングシステムのスケジューリング方法では、例えば、リアルタイムプロセスが何らかの原因でＣＰＵの占有権を放棄しなかったり、処理が所定時間内に完了しなかったりした場合に、計算機システムにデッドロックが発生したり、対話型プロセスが全く動作しなくなったり、応答時間が異常に遅くなる等の問題が生じた。本発明は上記問題点を解決するためになされたもので、その目的とするところは、リアルタイムプロセスが連続してＣＰＵを占有することを防ぐことにより動作を安定させることができる計算機システムにおけるプロセスのスケジューリング装置を提供することにある。

【００１３】

【課題を解決するための手段】上記目的を達成するために、第１の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したらプロセス待ち行列にリンクされている対話型プロセスの中の優先順位が最高の対話型プロセスにＣＰＵ占有を割り当てる手段とを備えたことを特徴とする。

【００１４】第２の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、ユーザにより作成されたハンドラプログラムを予め格納した記憶手段と、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したら記憶手段のハンドラプログラムによりスケジューリングを実行する手段とを備えたことを特徴とする。

【００１５】第３の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したら現在走行中のリアルタイムプロセスを強制終了するとともにプロセス待ち行列のリンクから外す手段とを備えたことを特徴とする。

【００１６】第４の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したら現在走行中のリアルタイムプロセスとプロセス待ち行列にリンクされている次の優先順位のリアルタイムプロセスとの間で互いの優先順位を入れ換える手段とを備えたことを特徴とする。

【００１７】第５の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、リアルタイムクラスのプロセスが連続してＣＰＵを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したらプロセス待ち行列にリンクされている対話型クラスの中の優先順位が最高の対話型プロセスにＣＰＵ占有を割り当てる手段とを備えたことを特徴とする。

【００１８】第６の発明は、プロセス待ち行列にリンクされた対話型プロセスおよびリアルタイムプロセスに対しそれぞれの優先順位に基づき定周期タイマ割込みによりＣＰＵ占有を割り当てる計算機システムにおけるプロセスのスケジューリング装置において、ユーザにより作成されたハンドラプログラムを予め格納した記憶手段と、リアルタイムクラスのプロセスが連続してＣＰＵを占有した回数をカウントする手段と、カウント手段のカウント値が所定値に達したら記憶手段のハンドラプログラムによりスケジューリングを実行する手段とを備えたことを特徴とする。

【００１９】

【作用】第１の発明においては、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数がカウントされそのカウント値が所定値に達すると、プロセス待ち行列にリンクされている対話型プロセスの中の優先順位が最高の対話型プロセスにＣＰＵ占有が割り当てられる。それにより、同一のリアルタイムプロセスが無制限に連続してＣＰＵを占有することがなくなる。

【００２０】第２の発明においては、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数がカウントされそのカウント値が所定値に達すると、予めユーザが作成しておいたハンドラプログラムにより以後のスケジューリングが実行される。それにより、同一のリアルタイムプロセスが連続してＣＰＵを占有した場合にユーザが任意の対処を行うことが可能になる。

【００２１】第３の発明においては、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数がカウントされそのカウント値が所定値に達すると、現在走行中のリアルタイムプロセスが強制終了されるとともにそのリアルタイムプロセスがプロセス待ち行列のリンクから外される。それにより、異常原因等により連続してＣＰＵ

を占有したリアルタイムプロセスが除去され、以後正常な動作に復帰する。

【００２２】第４の発明においては、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数がカウントされそのカウント値が所定値に達すると、現在走行中のリアルタイムプロセスとプロセス待ち行列にリンクされている次の優先順位のリアルタイムプロセスとの間で互いの優先順位が入れ換えられる。それにより、同一のリアルタイムプロセスが無制限に連続してＣＰＵを占有することがなくなる。

【００２３】第５の発明においては、リアルタイムクラスのプロセスが連続してＣＰＵを占有した回数がカウントされそのカウント値が所定値に達すると、プロセス待ち行列にリンクされている対話型クラスの中の優先順位が最高の対話型プロセスにＣＰＵ占有が割り当てられる。それにより、リアルタイムプロセスが無制限に連続してＣＰＵを占有することがなくなる。

【００２４】第６の発明においては、リアルタイムクラスのプロセスが連続してＣＰＵを占有した回数がカウントされそのカウント値が所定値に達すると、予めユーザが作成しておいたハンドラプログラムにより以後のスケジューリングが実行される。それにより、リアルタイムプロセスが連続してＣＰＵを占有した場合にユーザが任意の対処を行うことが可能になる。

【００２５】

【実施例】以下、図に沿って本発明の実施例を説明する。この実施例で対象とする計算機システムのオペレーティングシステムは、プロセスの優先順位に基づき、対話型プロセスとリアルタイムプロセスのＣＰＵへの割り当てを管理するものであり、以下の条件で作動する。

（１）定周期タイマが一定間隔（例えば、１０ｍｓ）でオペレーティングシステムに対し割込みを発生する。

（２）対話型プロセスとリアルタイムプロセスは互いに他者に対し、異なる優先順位を持つ。つまり、同種のプロセス間では同じ優先順位を持つことがあっても、対話型プロセスとリアルタイムプロセスとが同じ優先順位を持つことはない。

（３）ＣＰＵの割当てを待つためのプロセスの待ち行列を、優先順位ごとに持つ。

（４）各プロセスは、システムで唯一の識別子を持つ。

【００２６】図１は本発明が適用されるオペレーティングシステムの構成を示す概念図である。図示されるように、オペレーティングシステムはタイマ割込み処理モジュール、プロセス選択処理モジュール、プロセス切替え処理モジュール、対話型プロセス切替え処理モジュール、プロセス待ち行列、現在走行中プロセス情報、監視時間情報、ハンドラプログラム登録テーブルおよびリアルタイムプロセス抑止フラグから構成されている。ここで、定周期タイマの割込みを受け付けるタイマ割込み処理モジュールおよびプロセス切替え処理モジュールは、従来技術の項で示した図１４の動作と同じであるのでその詳細な説明は省略する。また、プロセス選択処理モジュールおよび対話型プロセス切替え処理モジュールが本発明の主なる特徴であり、詳細な動作は後述する。

【００２７】プロセス待ち行列は、プロセスごとの優先順位に対応したリンクターミナルを備え、ＣＰＵの割当てを待つリアルタイムプロセスおよび対話型プロセスをそれぞれ対応する優先順位のリンクターミナルにリンクする。図示例では、優先順位が０～１２７の１２８あり、優先順位は数値が小さいほど高くなる。このうち上位の０～６３の優先順位をリアルタイムクラスとしてリアルタイムプロセスの優先順位とする。また残り下位の６４～１２７の優先順位を対話型クラスとして対話型プロセスの優先順位とする。

【００２８】現在走行中プロセス情報は、プロセスの識別子、プロセスの優先順位、同一プロセス走行カウンタおよび同一クラスプロセス走行カウンタからなる。このプロセス識別子には、現在ＣＰＵが割当てられているプロセスの識別子が格納される。プロセスの優先順位には、現在ＣＰＵが割当てられているプロセスの優先順位が格納される。同一プロセス走行カウンタは、同一のプロセスに対して連続してＣＰＵが割当てられた時間が単位時間（定周期タイマ割込み１回あたりのＣＰＵ占有時間）の個数として格納され、ＣＰＵが異なるプロセスへ割当てられたとき、このカウンタはゼロクリアされる。

【００２９】同一クラスプロセス走行カウンタは、同一のクラスのプロセスに対して連続してＣＰＵが割り当てられた時間が単位時間の個数として格納され、ＣＰＵが異なるクラスのプロセスへ割り当てられときにこのカウンタはゼロクリアされる。監視時間情報は、リアルタイムプロセスが所定時間以上ＣＰＵを占有していないかどうかをチェックする際の基準値として用いられる。つまり、この監視時間情報に予めシステムに応じた値として上述した単位時間の個数ｎをセットしておき、同一プロセス走行カウンタまたは同一クラスプロセス走行カウンタのカウント値と監視時間情報の値を比較し、カウント値が監視時間情報の値に達したら異常事態の発生と見なす。

【００３０】ハンドラプログラム登録テーブルはリアルタイムプロセスが所定時間以上ＣＰＵを占有したときに実行されるハンドラプログラムのエントリアドレスを格納している。このハンドラプログラムはユーザによって予め登録される。リアルタイムプロセス抑止フラグはリアルタイムクラスのプロセスのＣＰＵへの割当てを抑止する。

【００３１】次に、各発明をフローチャートに基づき説明する。図２は、第１の発明に係るプロセス選択処理モジュールの第１の実施例を示すフローチャートである。このモジュールは、先ず、現在走行中のリアルタイムプロセスが連続してＣＰＵを占有している時間を知るた

め、現在走行中のプロセス情報にある同一プロセス走行カウンタに１を加算する（Ｓ２１）。次に、現在走行中のプロセスに関する情報を現在走行中プロセス情報から獲得し（Ｓ２６）、次いで、現在走行中のプロセスがリアルタイムの優先順位か否かを判別する（Ｓ２７）。リアルタイムの優先順位である場合は（Ｓ２７Ｙ）、さらに、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さい場合は（Ｓ２８Ｙ）、プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得する（Ｓ３１）。

【００３２】次に、待ち行列のプロセスがリアルタイム優先順位か否かを判別し（Ｓ３２）、リアルタイム優先順位である場合は（Ｓ３２Ｙ）、さらに、待ち行列のプロセスは現在走行中のプロセスより優先順位が高いか否かを判別する（Ｓ３３）。優先順位が高い場合（Ｓ３３Ｙ）、すなわち図１のプロセス行列待ちに、現在走行中のプロセスより優先順位の高いリアルタイムプロセスがリンクされていたら図１３のプロセス切換え処理モジュールを実行し（Ｓ３４）、次いで、現在走行中プロセス情報の同一プロセス走行カウンタを０にして終了する（Ｓ３７）。

【００３３】また、Ｓ２７でリアルタイム優先順位でない場合は、Ｓ３４のプロセス切換え処理モジュールへ進む。さらに、Ｓ２８で同一プロセス走行カウンタの値が監視時間情報の値よりも大きいか等しい場合は、図３の対話型プロセス切替えモジュールを実行し（Ｓ４４）、次いで、現在走行中プロセス情報の同一プロセス走行カウンタを０にして終了する（Ｓ４７）。またさらに、Ｓ３２で待ち行列のプロセスがリアルタイム優先順位でなくて対話型優先順位である場合、およびＳ３３で優先順位が高くない、つまり低い場合は処理を終了することにより、現在走行中のプロセスにそのままＣＰＵの占有を続けさせる。

【００３４】図３は、対話型プロセス切替えモジュールを示すフローチャートである。先ず、対話型クラスのプロセス待ち行列から最も優先順位の高いプロセスの情報を獲得する（Ｓ６１）。次に、得られたプロセスは対話型プロセスであるか否かを判別し、対話型プロセスでない場合は（Ｓ６２Ｎ）、処理を終了する。対話型プロセスである場合は（Ｓ６２Ｙ）、現在走行中のプロセスをプロセス待ち行列へリンクする（Ｓ６３）。なお、ここでリンクするプロセスが対話型プロセスである場合は、優先順位を再計算してからリンクする。

【００３５】次に、対話型クラスのプロセス待ち行列から最も優先順位の高いプロセスのリンクを外し（Ｓ６４）、次いで、待ち行列からリンクが外されたプロセスにＣＰＵを割り当て（Ｓ６５）、同時に現在走行中プロセス情報に、ＣＰＵを割り当てたプロセスの識別子と優先順位を設定して終了する（Ｓ６６）。これらの第１の実施例にかかる処理では、Ｓ２８，Ｓ４４の処理を実行することで、同一のリアルタイムプロセスによりＣＰＵが連続して占有される場合でも、同一プロセス走行カウンタにセットされたｎ回のＣＰＵ占有がなされた後に、対話型プロセスによるＣＰＵの占有が１回行われる。すなわち、ｎ＋１回の定周期割込みの間に少なくとも１回は対話型プロセスが実行されることになる。

【００３６】その結果、同一のリアルタイムプロセスが暴走したり、一定時間内に処理が完了しなくても、計算機システムがデッドロックしたり、対話型プロセスが全く動作しなくなったりするというようなことがなくなる。また、対話型プロセスから暴走等した同一のリアルタイムプロセスを強制終了させることもできる。さらに、同一のリアルタイムプロセスが暴走等しても、それを異常とみなさずにそのままリアルタイム処理を継続させたいというようなシステムにこの実施例は有効である。

【００３７】図４は、第１の発明に係るプロセス選択処理モジュールの第２の実施例を示すフローチャートである。このフローチャートは、図２のフローチャートにＳ２２，Ｓ４１，Ｓ４２を追加したものであり、他は図２と共通であるので共通部分の説明を省略して異なる部分についてのみ説明する。すなわち、Ｓ２１で、現在走行中のプロセス情報にある同一プロセス走行カウンタに１を加算した後、リアルタイムプロセス抑止フラグをチェックし、１がセットされていれば（Ｓ２２Ｙ）、そのままＳ２６へ進み、１がセットされていなければ（Ｓ２２Ｎ）、Ｓ４１へ進む。

【００３８】また、Ｓ２８で現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合（Ｓ２８Ｎ）、Ｓ４１へ進む。Ｓ４１では、リアルタイムプロセス抑止フラグをチェックし、１がセットされていれば（Ｓ４１Ｎ）、そのままＳ４４へ進み、１がセットされていなければ（Ｓ４１Ｙ）、Ｓ４２へ進み、リアルタイムプロセス抑止フラグに１をセットして、Ｓ４４へ進む。これら第２の実施例にかかる処理では、同一のリアルタイムプロセスによりＣＰＵがｎ回連続して占有されると、リアルタイムプロセス抑止フラグに１がセットされて対話型プロセスにＣＰＵの制御権が渡され、以後毎回とも図３の対話型プロセス切替えモジュールが実行されることになる。

【００３９】その結果、同一のリアルタイムプロセスが暴走したり、一定時間内に処理が完了しなくても、計算機システムがデッドロックしたり、対話型プロセスが全く動作しなくなることがなくなる。また、対話型プロセスから暴走等した同一のリアルタイムプロセスを強制終了させることもできる。さらに、同一のリアルタイムプロセスが暴走等したら、システムに異常が発生したものとみなし、リアルタイム処理をそのまま続行せずに何らかの原因調査を行いたいというようなシステムにこの実施例は有効である。

【００４０】図５は、第２の発明に係るプロセス選択処理モジュールの第３の実施例を示すフローチャートである。このフローチャートは、図２のＳ４４をＳ４５，Ｓ４６にかえたものであり、他は図２と共通であるので共通部分の説明を省略し異なる部分についてのみ説明する。すなわち、Ｓ２８で、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合（Ｓ２８Ｎ）、Ｓ４５へ進む。

【００４１】Ｓ４５では、ハンドラプログラム登録テーブルにハンドラプログラムのエントリアドレスが登録されているか否かを判別し、登録されていなければ（Ｓ４５Ｎ）、そのままＳ４７へ進む。登録されていれば（Ｓ４５Ｙ）、そのエントリアドレスからハンドラプログラムを呼び出して実行し（Ｓ４６）、Ｓ４７へ進む。これら第３の実施例にかかる処理では、同一のリアルタイムプロセスによりＣＰＵがｎ回連続して占有されると、ハンドラプログラムが呼びだされて実行される。このハンドラプログラムは予めユーザにより登録されているため、ＣＰＵの割当てのスケジューリングをユーザが自由に設定してユーザ固有の最適なシステムを構築することができる。

【００４２】図６は、第３の発明に係るプロセス選択処理モジュールの第４の実施例を示すフローチャートである。このフローチャートは、図２のＳ４４，Ｓ４７をＳ４８にかえたものであり、他は図２と共通であるので共通部分の説明を省略し異なる部分についてのみ説明する。すなわち、Ｓ２８で、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合（Ｓ２８Ｎ）、Ｓ４８へ進む。Ｓ４８では、現在走行中のリアルタイムプロセスのＣＰＵ占有を強制終了する。このとき、終了したリアルタイムプロセスを、待ち行列のリンクターミナルへ接続させない。次に、Ｓ３４のプロセス切換え処理モジュールを実行することにより、次の優先順位のプロセスにＣＰＵが割り当てられる。

【００４３】これら第４の実施例にかかる処理では、同一のリアルタイムプロセスによりＣＰＵがｎ回連続して占有されると、それまで走行中のリアルタイムプロセスが強制的に終了され、次に優先順位の高いプロセスにＣＰＵが割当てられることで同一リアルタイムプロセスによるＣＰＵの独占が回避される。その結果、同一のリアルタイムプロセスが暴走したり、一定時間内に処理が完了しなくても、計算機システムがデッドロックしたり、対話型プロセスが全く動作しなくなったりすることがなくなる。

【００４４】もっとも、対話型プロセスが動作するのは、リアルタイムプロセスが終了してからである。また、対話型プロセスから暴走等した同一のリアルタイムプロセスを強制終了させることもできる。さらに、同一のリアルタイムプロセスが暴走等したら、そのプロセスだけに異常が発生したとみなしてリアルタイム処理を継続させたいというようなシステムにこの実施例は有効である。

【００４５】図７は、第４の発明に係るプロセス選択処理モジュールの第５の実施例を示すフローチャートである。このフローチャートは、図２のＳ４４，Ｓ４７をＳ４９～Ｓ５２にかえたものであり、他は図２と共通であるので共通部分の説明を省略し異なる部分についてのみ説明する。すなわち、Ｓ２８で、現在走行中プロセス情報中の同一プロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合（Ｓ２８Ｎ）、Ｓ４９へ進む。Ｓ４９では、リアルタイムプロセスの待ち行列から最も優先順位の高いプロセスの情報を獲得する。

【００４６】次いで、獲得した情報はリアルタイムプロセスか否かを判別し、リアルタイムプロセスでない場合は（Ｓ５０Ｎ）、処理を終了する。リアルタイムプロセスである場合は（Ｓ５０Ｙ）、現在走行中のリアルタイムプロセスとリアルタイムクラスのプロセス待ち行列の中で最も優先順位の高いプロセスとの互いの優先順位を交換する（Ｓ５１）。次に、現在走行中のリアルタイムプロセスをリアルタイムクラスのプロセス待ち行列にリンクし（Ｓ５２）、次いで、Ｓ３４のプロセス切換え処理モジュールを実行することにより、次の優先順位のリアルタイムプロセスにＣＰＵが割り当てられる。

【００４７】これら第５の実施例にかかる処理では、同一のリアルタイムプロセスによりＣＰＵがｎ回連続して占有されると、それまで走行中のリアルタイムプロセスと次に優先順位の高いリアルタイムプロセスとの優先順位を交換し、それまで次の優先順位であったリアルタイムプロセスにＣＰＵを割り当てることで、同一のリアルタイムプロセスが暴走したり、一定時間内に処理が完了しなくても計算機システムがデッドロックしたり、対話型プロセスが全く動作しなくなったりすることがなくなる。

【００４８】もっとも、対話型プロセスが動作するのは、リアルタイムプロセスが終了してからである。また、同一のリアルタイムプロセスが暴走等しても、それを異常とみなさず、しかも対話型プロセスにＣＰＵの占有権を渡さずに、リアルタイム処理は継続したいというようなシステムにこの実施例は有効である。また、この実施例の場合、暴走等した同一のリアルタイムプロセスを強制終了等させようとすると、次に優先順位の高いプロセスの処理が重要となる。

【００４９】図８は、第５の発明に係るプロセス選択処理モジュールの第６の実施例を示すフローチャートである。このモジュールは、先ず、現在走行中のリアルタイムクラスのプロセスが連続してＣＰＵを占有している時間を知るため、現在走行中のプロセス情報にある同一クラスプロセス走行カウンタに１を加算する（Ｓ２３）。次に、現在走行中のプロセスに関する情報を現在走行中

プロセス情報から獲得し（Ｓ２６）、次いで、現在走行中のプロセスがリアルタイムの優先順位か否かを判別する（Ｓ２７）。

【００５０】リアルタイムの優先順位である場合は（Ｓ２７Ｙ）、さらに、現在走行中プロセス情報中の同一クラスプロセス走行カウンタの値を監視時間情報の値と比較し、小さい場合は（Ｓ２９Ｙ）、プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得する（Ｓ３１）。次に、待ち行列のプロセスがリアルタイム優先順位か否かを判別し（Ｓ３２）、リアルタイム優先順位である場合は（Ｓ３２Ｙ）、さらに、待ち行列のプロセスは現在走行中のプロセスより優先順位が高いか否かを判別する（Ｓ３３）。

【００５１】優先順位が高い場合（Ｓ３３Ｙ）、すなわち図１のプロセス行列待ちに、現在走行中のプロセスより優先順位の高いリアルタイムプロセスがリンクされていたら図１３のプロセス切換え処理モジュールを実行する（Ｓ３４）。ここでさらに、以前走行していたプロセスと現在走行中のプロセスのクラスが違うが否かを判別し、違う場合は（Ｓ３５Ｙ）、現在走行中プロセス情報の同一クラスプロセス走行カウンタを０にして終了する（Ｓ３７）。また、Ｓ２７でリアルタイム優先順位でない場合は、プロセス切換え処理モジュール（Ｓ３４）へ進む。

【００５２】さらに、Ｓ２９で同一クラスプロセス走行カウンタの値が監視時間情報の値よりも大きいか等しい場合は、図３の対話型プロセス切替えモジュールを実行し（Ｓ４４）、以前走行していたプロセスと現在走行中のプロセスのクラスが違うが否かを判別し、違う場合は（Ｓ５３Ｙ）、現在走行中プロセス情報の同一クラスプロセス走行カウンタを０にして終了する（Ｓ５４）。また、プロセスのクラスが同じ場合は（Ｓ５３Ｎ）、そのまま終了する。またさらに、Ｓ３２で待ち行列のプロセスがリアルタイム優先順位でなくて対話型優先順位である場合、Ｓ３３で優先順位が高くない場合およびＳ３５で以前走行していたプロセスと現在走行中のプロセスのクラスが同じ場合は処理を終了することにより、現在走行中のプロセスがそのままＣＰＵの占有を続ける。

【００５３】これらの第６の実施例にかかる処理では、Ｓ２９，Ｓ４４の処理を実行することで、リアルタイムクラスのプロセスによりＣＰＵが連続して占有される場合でも、同一クラスプロセス走行カウンタにセットされたｎ回のＣＰＵ占有がなされた後に、対話型プロセスによるＣＰＵの占有が１回行われる。すなわち、ｎ＋１回の定周期割込みの間に少なくとも１回は対話型プロセスが実行されることになる。また、第６の実施例は、第１の実施例とほぼ同じ効果が得られるとともに、対話型プロセスの応答時間が異常に遅くなることも解消される。

【００５４】図９は、第５の発明に係るプロセス選択処理モジュールの第７の実施例を示すフローチャートである。このフローチャートは、図８のフローチャートにＳ２４，Ｓ４１，Ｓ４２を追加したものであり、他は図８と共通であるので共通部分の説明を省略して異なる部分についてのみ説明する。すなわち、Ｓ２３で、同一クラスプロセス走行カウンタに１を加算した後、リアルタイムプロセス抑止フラグをチェックし、１がセットされていれば（Ｓ２４Ｙ）、そのままＳ２６へ進み、１がセットされていなければ（Ｓ２４Ｎ）、Ｓ４１へ進む。また、Ｓ２９で現在走行中プロセス情報中の同一クラスプロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合（Ｓ２９Ｎ）、Ｓ４１へ進む。

【００５５】Ｓ４１では、リアルタイムプロセス抑止フラグをチェックし、１がセットされていれば（Ｓ４１Ｎ）、そのままＳ４４へ進む。１がセットされていなければ（Ｓ４１Ｙ）、Ｓ４２へ進み、リアルタイムプロセス抑止フラグに１をセットして、Ｓ４４へ進む。これら第７の実施例にかかる処理では、リアルタイムクラスのプロセスによりＣＰＵがｎ回連続して占有されると、リアルタイムプロセス抑止フラグに１がセットされて、以後は毎回とも図３の対話型プロセス切替えモジュールが実行されることになる。また、第７の実施例は、第２の実施例とほぼ同じ効果が得られるとともに、対話型プロセスの応答時間が異常に遅くなることも解消される。

【００５６】図１０は、第６の発明に係るプロセス選択処理モジュールの第８の実施例を示すフローチャートである。このフローチャートは、図８のＳ４４をＳ４５，Ｓ４６にかえたものであり、他は図８と共通であるので共通部分の説明を省略し異なる部分についてのみ説明する。すなわち、Ｓ２９で、現在走行中プロセス情報中の同一クラスプロセス走行カウンタの値を監視時間情報の値と比較し、小さくない場合（Ｓ２９Ｎ）、Ｓ４５へ進む。Ｓ４５では、ハンドラプログラム登録テーブルにハンドラプログラムのエントリアドレスが登録されているか否かを判別し、登録されていなければ（Ｓ４５Ｎ）、そのままＳ５３へ進む。登録されていれば（Ｓ４５Ｙ）、そのエントリアドレスからハンドラプログラムを呼び出して実行して（Ｓ４６）、Ｓ５３へ進む。

【００５７】これら第８の実施例にかかる処理では、リアルタイムクラスのプロセスによりＣＰＵがｎ回連続して占有されると、ハンドラプログラムを呼びだして実行される。このハンドラプログラムは予めユーザにより登録されているため、ＣＰＵの割当てのスケジューリングをユーザが自由に設定してユーザ固有の最適なシステムを構築することができる。また、第８の実施例は、第３の実施例とほぼ同じ効果が得られるとともに、対話型プロセスの応答時間が異常に遅くなることも解消される。

【００５８】

【発明の効果】以上述べたように第１の発明によれば、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数が所定回数になると、プロセス待ち行列にリンク

されている対話型プロセスの中の優先順位が最高の対話型プロセスにＣＰＵの占有を割り当てることにより、同一のリアルタイムプロセスが無制限に連続してＣＰＵを占有することがなくなる。

【００５９】第２の発明によれば、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数が所定回数になると、予めユーザが作成しておいたハンドラプログラムにより以後のスケジューリングを実行することにより、同一のリアルタイムプロセスが連続してＣＰＵを占有した場合にユーザが任意の対処を行うことが可能になる。

【００６０】第３の発明によれば、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数が所定回数になると、現在走行中のリアルタイムプロセスを強制終了させるとともにそのリアルタイムプロセスをプロセス待ち行列のリンクから外すことにより、異常原因等により連続してＣＰＵを占有したリアルタイムプロセスが除去され、以後正常な動作に復帰する。

【００６１】第４の発明によれば、同一のリアルタイムプロセスが連続してＣＰＵを占有した回数が所定回数になると、現在走行中のリアルタイムプロセスとプロセス待ち行列にリンクされている次の優先順位のリアルタイムプロセスとの間で互いの優先順位を入れ換えることにより、同一のリアルタイムプロセスが無制限に連続してＣＰＵを占有することがなくなる。

【００６２】第５の発明によれば、リアルタイムクラスのプロセスが連続してＣＰＵを占有した回数が所定回数になると、プロセス待ち行列にリンクされている対話型クラスの中の優先順位が最高の対話型プロセスにＣＰＵ占有を割り当てることにより、リアルタイムプロセスが無制限に連続してＣＰＵを占有することがなくなる。

【００６３】第６の発明によれば、リアルタイムクラスのプロセスが連続してＣＰＵを占有した回数が所定回数になると、予めユーザが作成しておいたハンドラプログラムにより以後のスケジューリングを実行することにより、リアルタイムプロセスが連続してＣＰＵを占有した場合にユーザが任意の対処を行うことが可能になる。

【００６４】これらのことから各発明については、次のような効果が得られる。

（１）同一または同一クラスのリアルタイムプロセスの暴走を防止できる。

（２）一定時間内に処理が終了しない場合でも、計算機システムがデッドロックにおちいることを防止できる。

（３）同じく、一定時間内に処理が終了しない場合でも、対話型プロセスが全く作動しないという事態が避けられる。

（４）対話型プロセスから暴走等を起こした同一のリアルタイムプロセスを強制終了させることも可能である。

（５）さらには、同一のリアルタイムプロセスが暴走等を起こしても、それを異常とみなすことなくそのままリアルタイム処理を継続することも可能である。

【図面の簡単な説明】

【図１】本発明が適用されるオペレーティングシステムの構成を示す概念図である。

【図２】第１の発明に係る第１の実施例を示すフローチャートである。

【図３】図２の要部を示すフローチャートである。

【図４】第１の発明に係る第２の実施例を示すフローチャートである。

【図５】第２の発明に係る第３の実施例を示すフローチャートである。

【図６】第３の発明に係る第４の実施例を示すフローチャートである。

【図７】第４の発明に係る第５の実施例を示すフローチャートである。

【図８】第５の発明に係る第６の実施例を示すフローチャートである。

【図９】第５の発明に係る第７の実施例を示すフローチャートである。

【図１０】第６の発明に係る第８の実施例を示すフローチャートである。

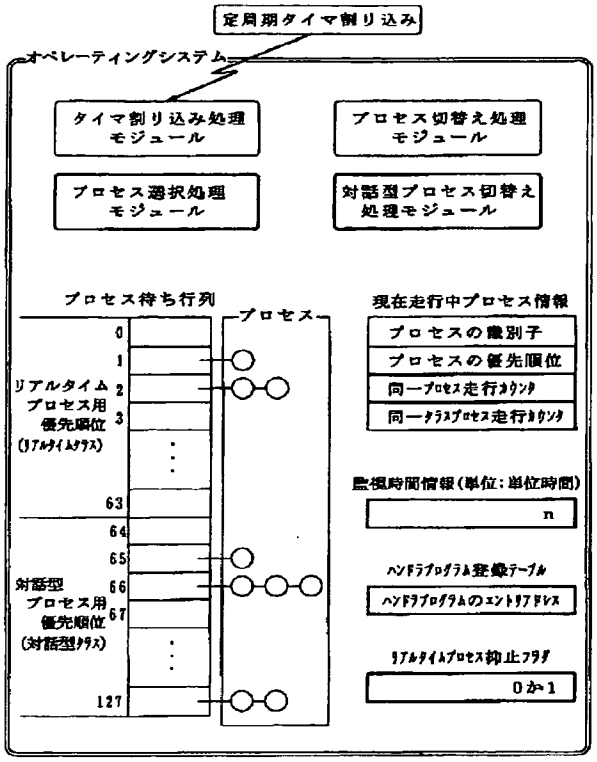
【図１１】従来のスケジューリング処理を示すフローチャートである。

【図１２】従来のスケジューリング処理を示すフローチャートである。

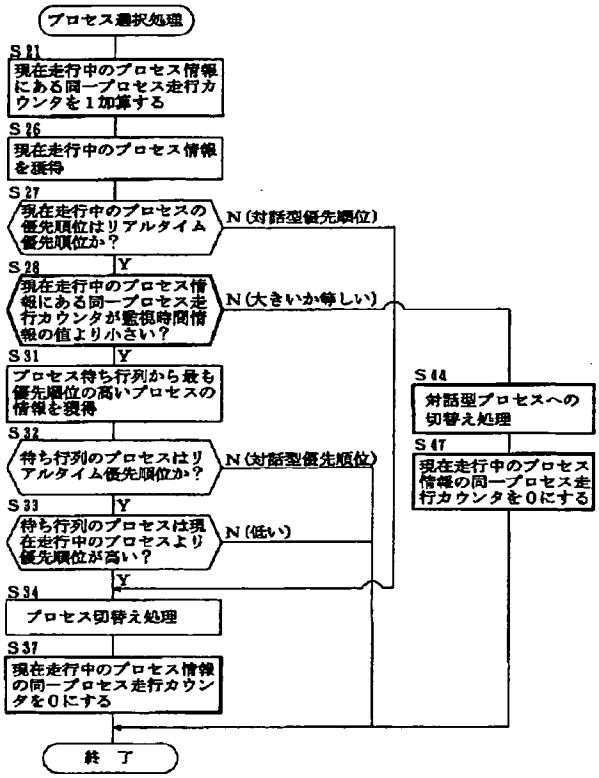
【図１３】従来のスケジューリング処理を示すフローチャートである。

【図１４】従来のオペレーティングシステムの構成を示した概念図である。

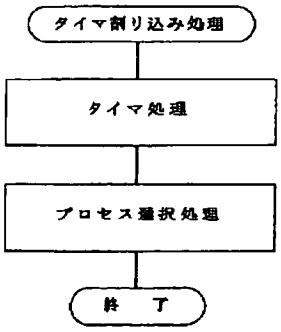
【図１】



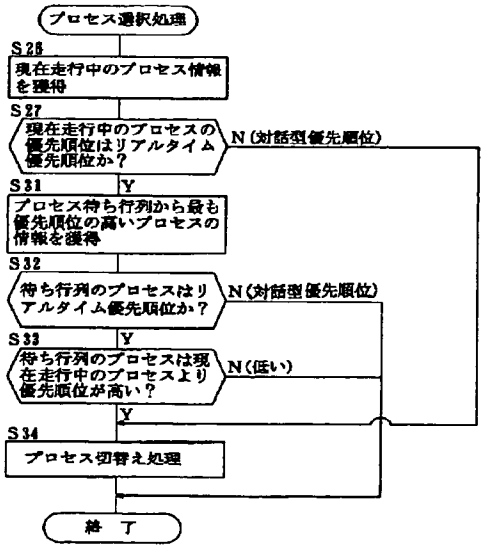
【図２】



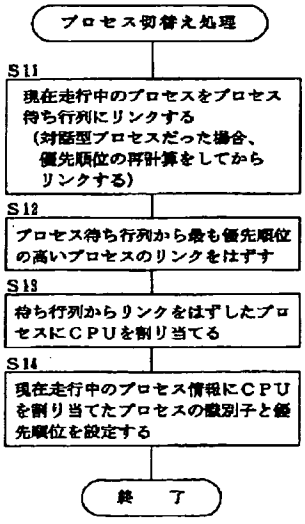
【図１１】



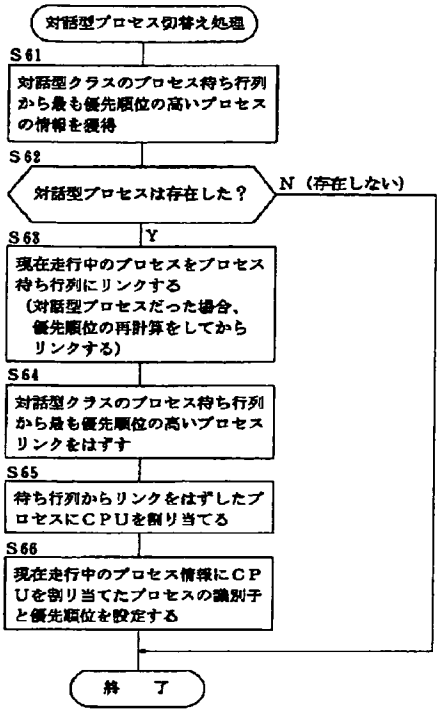
【図１２】



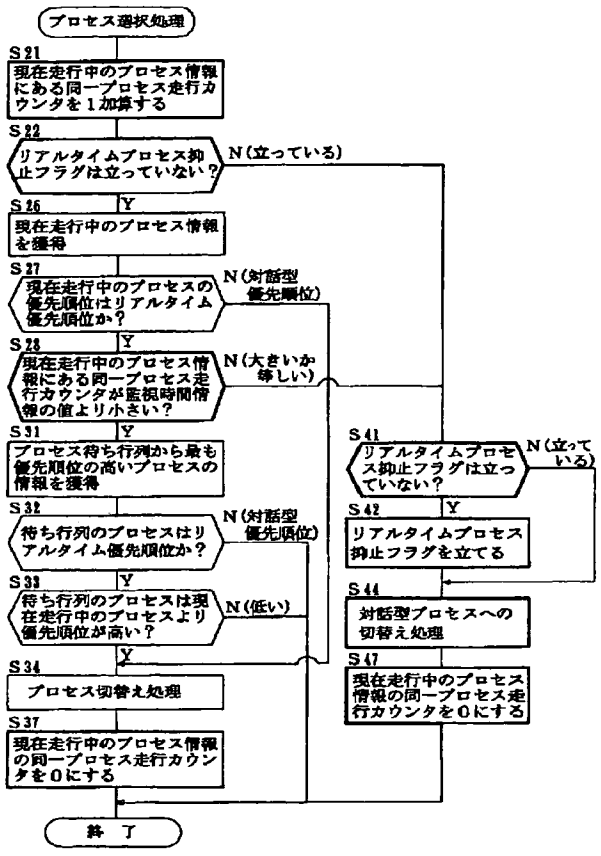
【図１３】



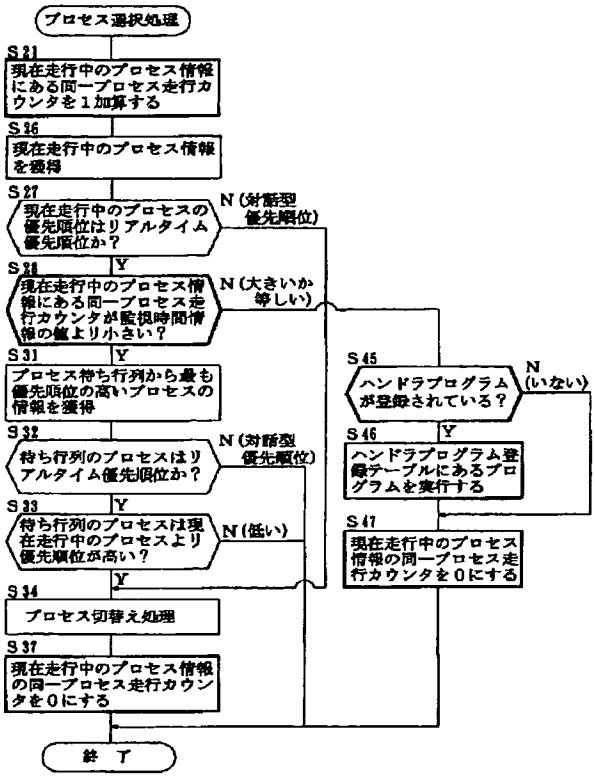
【図３】


対話型プロセス切替え処理
S61
対話型クラスのプロセス待ち行列から最も優先順位の高いプロセスの情報を獲得
S62
対話型プロセスは存在した？
N（存在しない）
Y
S63
現在走行中のプロセスをプロセス待ち行列にリンクする（対話型プロセスだった場合、優先順位の再計算をしてからリンクする）
S64
対話型クラスのプロセス待ち行列から最も優先順位の高いプロセスリンクをはずす
S65
待ち行列からリンクをはずしたプロセスにＣＰＵを割り当てる
S66
現在走行中のプロセス情報にＣＰＵを割り当てたプロセスの識別子と優先順位を設定する
終　了

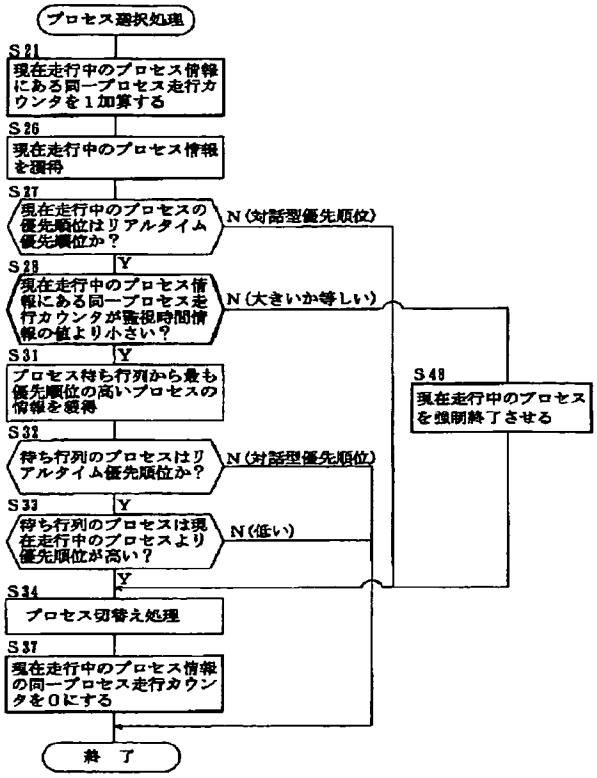
【図４】


プロセス選択処理
S21
現在走行中のプロセス情報にある同一プロセス走行カウンタを１加算する
S22
リアルタイムプロセス抑止フラグは立っていない？
N（立っている）
Y
S26
現在走行中のプロセス情報を獲得
S27
現在走行中のプロセスの優先順位はリアルタイム優先順位か？
N（対話型優先順位）
Y
S28
現在走行中のプロセス情報にある同一プロセス走行カウンタが監視時間情報の値より小さい？
N（大きいか等しい）
Y
S31
プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得
S32
待ち行列のプロセスはリアルタイム優先順位か？
N（対話型優先順位）
Y
S33
待ち行列のプロセスは現在走行中のプロセスより優先順位が高い？
N（低い）
Y
S34
プロセス切替え処理
S37
現在走行中のプロセス情報の同一プロセス走行カウンタを０にする
S41
リアルタイムプロセス抑止フラグは立っていない？
N（立っている）
Y
S42
リアルタイムプロセス抑止フラグを立てる
S44
対話型プロセスへの切替え処理
S47
現在走行中のプロセス情報の同一プロセス走行カウンタを０にする
終　了

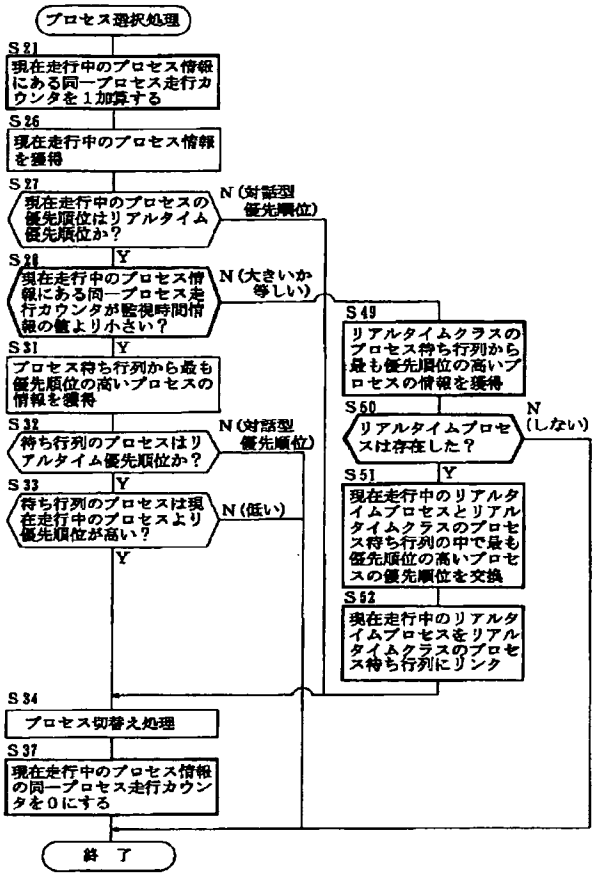
【図５】


プロセス選択処理
S21
現在走行中のプロセス情報にある同一プロセス走行カウンタを１加算する
S26
現在走行中のプロセス情報を獲得
S27
現在走行中のプロセスの優先順位はリアルタイム優先順位か？
N（対話型優先順位）
Y
S28
現在走行中のプロセス情報にある同一プロセス走行カウンタが監視時間情報の値より小さい？
N（大きいか等しい）
Y
S31
プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得
S32
待ち行列のプロセスはリアルタイム優先順位か？
N（対話型優先順位）
Y
S33
待ち行列のプロセスは現在走行中のプロセスより優先順位が高い？
N（低い）
Y
S34
プロセス切替え処理
S37
現在走行中のプロセス情報の同一プロセス走行カウンタを０にする
終　了
S45
ハンドラプログラムが登録されている？
N（いない）
Y
S46
ハンドラプログラム登録テーブルにあるプログラムを実行する
S47
現在走行中のプロセス情報の同一プロセス走行カウンタを０にする

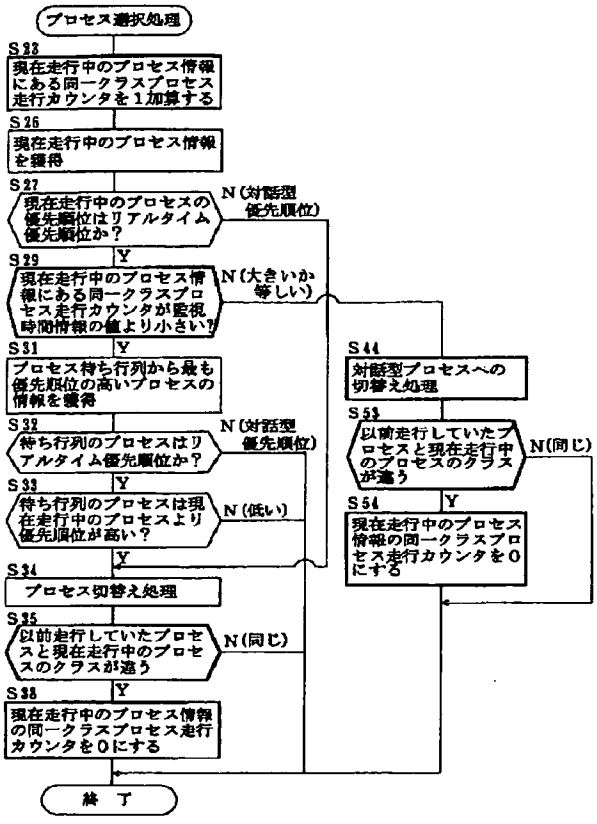
【図６】


プロセス選択処理
S21
現在走行中のプロセス情報にある同一プロセス走行カウンタを１加算する
S26
現在走行中のプロセス情報を獲得
S27
現在走行中のプロセスの優先順位はリアルタイム優先順位か？
N（対話型優先順位）
Y
S28
現在走行中のプロセス情報にある同一プロセス走行カウンタが監視時間情報の値より小さい？
N（大きいか等しい）
Y
S31
プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得
S32
待ち行列のプロセスはリアルタイム優先順位か？
N（対話型優先順位）
Y
S33
待ち行列のプロセスは現在走行中のプロセスより優先順位が高い？
N（低い）
Y
S34
プロセス切替え処理
S37
現在走行中のプロセス情報の同一プロセス走行カウンタを０にする
終　了
S48
現在走行中のプロセスを強制終了させる

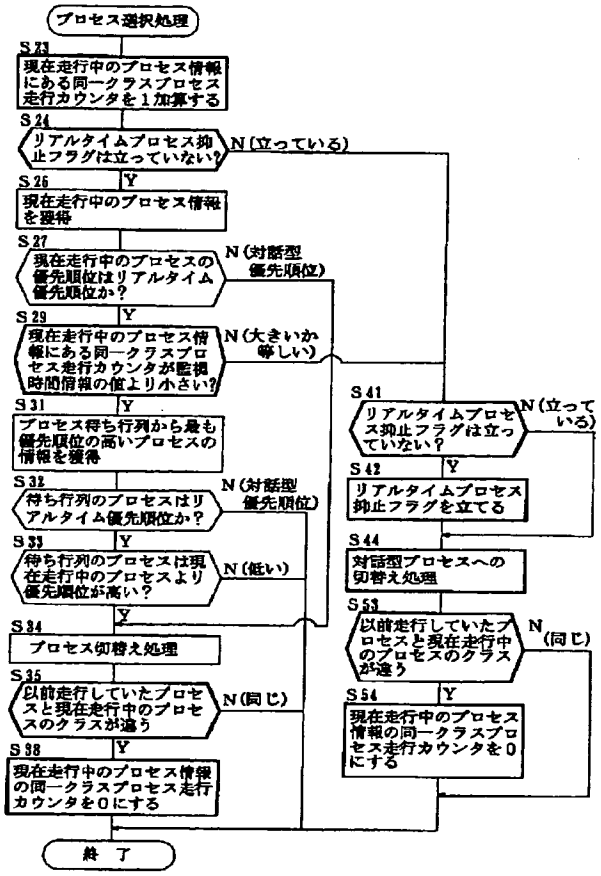
【図７】


プロセス選択処理
S21
現在走行中のプロセス情報にある同一プロセス走行カウンタを１加算する
S26
現在走行中のプロセス情報を獲得
S27
現在走行中のプロセスの優先順位はリアルタイム優先順位か？
N（対話型優先順位）
Y
S28
現在走行中のプロセス情報にある同一プロセス走行カウンタが監視時間情報の値より小さい？
N（大きいか等しい）
Y
S31
プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得
S32
待ち行列のプロセスはリアルタイム優先順位か？
N（対話型優先順位）
Y
S33
待ち行列のプロセスは現在走行中のプロセスより優先順位が高い？
N（低い）
Y
S34
プロセス切替え処理
S37
現在走行中のプロセス情報の同一プロセス走行カウンタを０にする
終　了
S49
リアルタイムクラスのプロセス待ち行列から最も優先順位の高いプロセスの情報を獲得
S50
リアルタイムプロセスは存在した？
N（しない）
Y
S51
現在走行中のリアルタイムプロセスとリアルタイムクラスのプロセス待ち行列の中で最も優先順位の高いプロセスの優先順位を交換
S52
現在走行中のリアルタイムプロセスをリアルタイムクラスのプロセス待ち行列にリンク

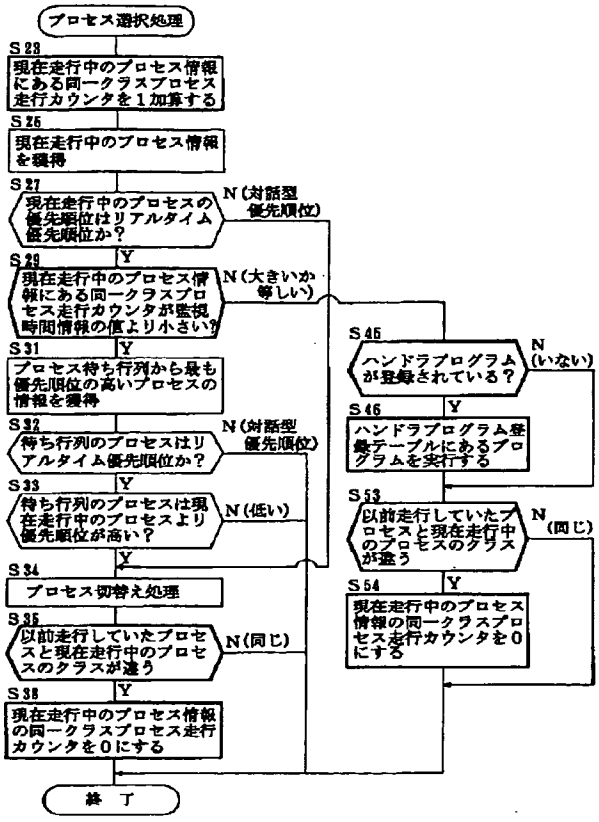
【図８】


プロセス選択処理
S23
現在走行中のプロセス情報にある同一クラスプロセス走行カウンタを１加算する
S26
現在走行中のプロセス情報を獲得
S27
現在走行中のプロセスの優先順位はリアルタイム優先順位か？
N（対話型優先順位）
Y
S29
現在走行中のプロセス情報にある同一クラスプロセス走行カウンタが監視時間情報の値より小さい？
N（大きいか等しい）
Y
S31
プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得
S32
待ち行列のプロセスはリアルタイム優先順位か？
N（対話型優先順位）
Y
S33
待ち行列のプロセスは現在走行中のプロセスより優先順位が高い？
N（低い）
Y
S34
プロセス切替え処理
S35
以前走行していたプロセスと現在走行中のプロセスのクラスが違う
N（同じ）
Y
S38
現在走行中のプロセス情報の同一クラスプロセス走行カウンタを０にする
終　了
S44
対話型プロセスへの切替え処理
S53
以前走行していたプロセスと現在走行中のプロセスのクラスが違う
N（同じ）
Y
S54
現在走行中のプロセス情報の同一クラスプロセス走行カウンタを０にする

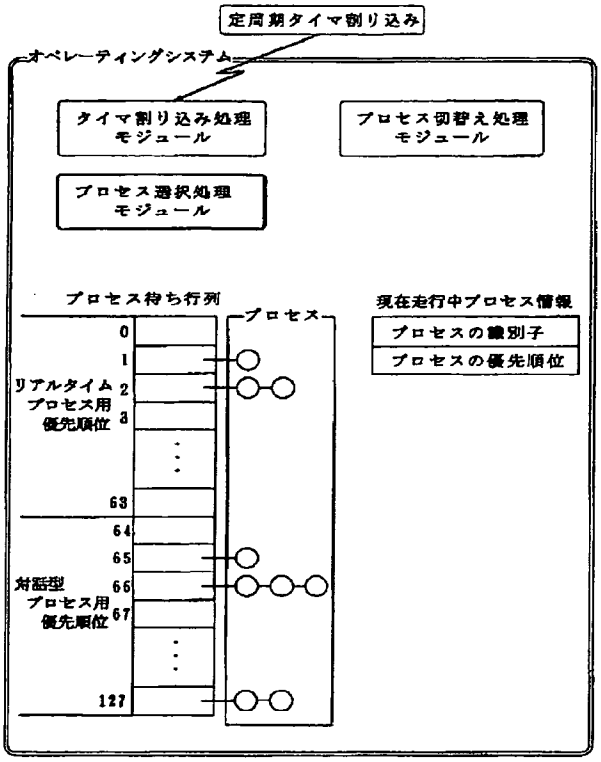
【図９】


プロセス選択処理
S23
現在走行中のプロセス情報にある同一クラスプロセス走行カウンタを１加算する
S24
リアルタイムプロセス抑止フラグは立っていない?
N(立っている)
Y
S26
現在走行中のプロセス情報を獲得
S27
現在走行中のプロセスの優先順位はリアルタイム優先順位か?
N(対話型優先順位)
Y
S29
現在走行中のプロセス情報にある同一クラスプロセス走行カウンタが監視時間情報の値より小さい?
N(大きいか等しい)
Y
S31
プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得
S32
待ち行列のプロセスはリアルタイム優先順位か?
N(対話型優先順位)
Y
S33
待ち行列のプロセスは現在走行中のプロセスより優先順位が高い?
N(低い)
Y
S34
プロセス切替え処理
S35
以前走行していたプロセスと現在走行中のプロセスのクラスが違う
N(同じ)
Y
S38
現在走行中のプロセス情報の同一クラスプロセス走行カウンタを０にする
終了
S41
リアルタイムプロセス抑止フラグは立っていない?
N(立っている)
Y
S42
リアルタイムプロセス抑止フラグを立てる
S44
対話型プロセスへの切替え処理
S53
以前走行していたプロセスと現在走行中のプロセスのクラスが違う
N(同じ)
Y
S54
現在走行中のプロセス情報の同一クラスプロセス走行カウンタを０にする

【図１０】


プロセス選択処理
S23
現在走行中のプロセス情報にある同一クラスプロセス走行カウンタを１加算する
S26
現在走行中のプロセス情報を獲得
S27
現在走行中のプロセスの優先順位はリアルタイム優先順位か?
N(対話型優先順位)
Y
S29
現在走行中のプロセス情報にある同一クラスプロセス走行カウンタが監視時間情報の値より小さい?
N(大きいか等しい)
Y
S31
プロセス待ち行列から最も優先順位の高いプロセスの情報を獲得
S32
待ち行列のプロセスはリアルタイム優先順位か?
N(対話型優先順位)
Y
S33
待ち行列のプロセスは現在走行中のプロセスより優先順位が高い?
N(低い)
Y
S34
プロセス切替え処理
S35
以前走行していたプロセスと現在走行中のプロセスのクラスが違う
N(同じ)
Y
S38
現在走行中のプロセス情報の同一クラスプロセス走行カウンタを０にする
終了
S45
ハンドラプログラムが登録されている?
N(いない)
Y
S46
ハンドラプログラム登録テーブルにあるプログラムを実行する
S53
以前走行していたプロセスと現在走行中のプロセスのクラスが違う
N(同じ)
Y
S54
現在走行中のプロセス情報の同一クラスプロセス走行カウンタを０にする

【図１４】


定周期タイマ割り込み
オペレーティングシステム
タイマ割り込み処理モジュール
プロセス切替え処理モジュール
プロセス選択処理モジュール
プロセス待ち行列
プロセス
現在走行中プロセス情報
プロセスの識別子
プロセスの優先順位
リアルタイムプロセス用優先順位
対話型プロセス用優先順位
0
1
2
3
63
64
65
66
67
127